A Fast 2-Approximation of Minimum Manhattan Networks

Bernhard Fuchs[a], Anna Schulze[b]

[a] *Abteilung Algorithmik, TU Braunschweig, Germany*
[b] *Zentrum für Angewandte Informatik Köln, Universität Köln, Germany*

Abstract

Given a set P of n points in the plane, a *Manhattan network* of P is a network that contains a rectilinear shortest path between every pair of points of P. A *minimum Manhattan network* of P is a Manhattan network of minimum total length.
The minimum Manhattan network problem is strongly NP-complete [4]. The best approximations published so far are an LP-based 2-approximation algorithm [3], and two combinatorial 2-approximation algorithms in time $O(n^2)$ and $O(n \log n)$ [10, 11]. We present a new combinatorial 2-approximation for this problem in time $O(n \log n)$.

Key words: minimum Manhattan networks, approximation algorithms, rectilinear routing, VLSI design, 1-spanner

1. Introduction

Connecting a given set of points with minimum total length is a key problem in VLSI design. In the routing process a set of components, e. g., transistors or gates, have to be connected by wires. The wires are allowed to run in two perpendicular directions. The wire length significantly affects the power consumption and the time to spread the signal across the chip. Minimum Steiner trees minimize the total wire length. Manhattan networks impose an additional constraint. In contrast to Steiner trees they must contain a *shortest* path between each pair of points. This reflects the requirement to transmit signals between pairs of components on the chip fast. Manhattan networks are defined in the *rectilinear metric* where one is allowed to use only horizontal and vertical lines. A *minimum Manhattan network* is a Manhattan network of minimum total length. See Figure 1 (a) and (b) for an example.

Another application of Manhattan networks is described by Lam et al. [16], who use a variant of the Manhattan network problem to align gene sequences. They ask only for certain node pairs to be connected by shortest paths and solve the problem by a modification of an algorithm of Gudmundsson et al. [9].

Manhattan networks fit in the concept of spanners. Given a set P of points in the plane, a given metric, and a number $t \in \mathbb{R}$ with $t \geq 1$, a network is a t-spanner for P under the given metric, if for each pair of points $p, q \in P$, there exists a (p, q)-path in the network of length at most t times the distance between p and q under the appropriate norm. A Manhattan network is a 1-spanner in the rectilinear metric. Spanners are commonly known and first introduced for the Euclidean metric by Chew [5]. They are studied extensively, see for instance the survey of Eppstein [6] or the book of Narasimhan and Smid [14]. Note that the Euclidean 1-spanner is the trivial complete graph.

Email addresses: fuchs@ibr.cs.tu-bs.de (Bernhard Fuchs), schulze@zpr.uni-koeln.de (Anna Schulze)

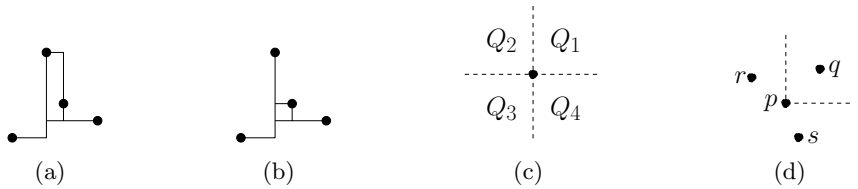


Figure 1: (a) A Manhattan network. (b) A minimum Manhattan network. (c) The quadrants of p. (d) Global and local neighborhood.

Gudmundsson et al. [8] has shown that there is a Manhattan network with $O(n \log n)$ vertices and edges which can be constructed in $O(n \log n)$ time. Just recently, Chin et al. [4] has shown the minimum Manhattan network problem to be strongly NP-complete. Muñoz et al. [13] proved that the MMN problem in three dimensions is APX-hard and there is no PTAS algorithm unless P=NP. For two dimensions it is not known whether a PTAS exists. Gudmundsson et al. [9] introduced an 8-approximation with running time $O(n \log n)$ and a 4-approximation running in time $O(n^3)$. Benkert et al. [1] gave a 3-approximation with running time $O(n \log n)$. We presented also a 3-approximation algorithm with the same running time [7]. Kato et al. [12] proposed a 2-approximation with running time $O(n^3)$, however the proof of the correctness seems to be incomplete [1]. Chepoi et al. [3] presented a 2-approximation based on LP-rounding. Their LP consists of $O(n^3)$ variables and constraints. In his PhD thesis (written in French) Nouioua [15] gave a 2-approximation that runs in $O(n \log n)$. Based on the primal-dual method, in contrast to the approach of Chepoi et al. the algorithm avoids to solve an LP. This work is unpublished until now. Guo et al. [10] presented first a 2-approximation with running time $O(n^2)$ and improve this to $O(n \log n)$ [11]. Unfortunately, in the approximation analysis of [11] is a mistake as we will describe in Section 3. Seibert and Unger [17] presented an approximation algorithm and claimed that it yields a 1.5-approximation. As remarked by Chepoi et al. [3] the description of both the algorithm and the performance guarantee are somewhat incomplete and not fully understandable.

We present a 2-approximation for minimum Manhattan networks with a running time of $O(n \log n)$. Our algorithm is somewhat similar to the algorithm of Benkert et al. [1]. We will discuss similarities and differences to their and also to other related algorithms below.

2. Definitions

We denote by p_x the x- and by p_y the y-coordinates of a point p. Each pair of points p and q spans a unique closed axis-parallel rectangle $R(p, q)$ with p and q as corners. By $\partial R(p, q)$ we denote the boundary of $R(p, q)$. As in [17], we call $R(p, q)$ *critical* if it does not contain any other point of P. To get shortest paths it suffices to consider critical rectangles: If a rectangle $R(p, q)$ contains a further point $r \in P$ then it suffices to consider the two rectangles $R(p, r)$ and $R(r, q)$. For a point $p \in \mathbb{R}^2$ we denote by $Q_1(p), \ldots, Q_4(p)$ the four open quadrants of p. See also Figure 1 (c).

We call two points $p, q \in P$ ($p_x \leq q_x$ and $p_y \leq q_y$) *x-neighboring* if there is no further point $r \in P$ with $p_x < r_x < q_x$ and *y-neighboring* if there is no further point $r \in P$ with $p_y < r_y < q_y$. Two x- or y-neighboring points form a *strip rectangle* as defined in several papers, i. e., [1, 3, 11].

For two points of P having the same x- or y-coordinate each Manhattan network must contain the direct line connecting these two points. Generally, the problem becomes easier in this case. Since we would have to do some rather technically distinctions concerning this case, in the following we assume the coordinates of all points to be different.

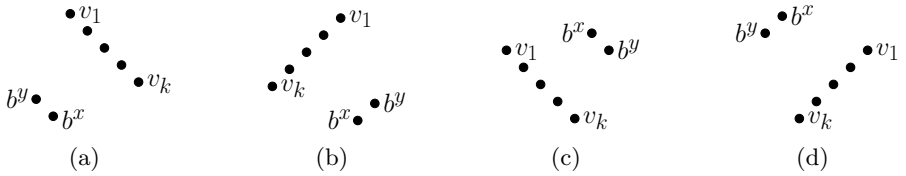


Figure 2: The four different cases of staircases.

We sometimes consider local neighborhoods. More precisely, for a point $p \in P$ we consider the x- or y-neighboring point in one of its quadrants. See for example Figure 1 (d). The x-neighboring point of p in the first quadrant of p is the point q, whereas the globally x-neighboring points of p are the points r and s. Nevertheless, if not stated otherwise we consider global neighborhoods.

Almost all approximation algorithms for minimum Manhattan networks use staircases, but the definition of a staircase is not standardized. We define as a staircase the same point sets as for example in [3, 10, 11]. To get a clearer definition we define only one of four symmetric cases of a staircase shown in Figure 2. We define the staircase type as shown in Figure 2 (a).

Definition 1. *For each point $p \in P$ the x-base point b^x is the x-neighboring point in $Q_3(p)$. The y-base point b^y is the y-neighboring point in $Q_3(p)$. Two points belong to the same* staircase *if they have the same base points.*

We also consider sequences with only one point as a staircase sequence.

To make the definition of a staircase clear, we first give two important properties of staircases which clarify their structure.

Observation 2. *Each sequence point forms a critical rectangle with each base point.*

In the following we assume the sequence points $(v_1, \ldots, v_k)$ sorted by increasing x-coordinate. We denote the points v_1 and v_k of a staircase sequence $(v_1, \ldots, v_k)$ as the *outer points* of the sequence. The next remark displays the typical structure of a staircase as depicted in Figure 2 (a).

Observation 3. *The set of points $(v_1, \ldots, v_k)$ with the same set of base points form a sequence which is monotone increasing in x- and monotone decreasing in y-direction.*

Let c be the point $c = (v_{1_x}, v_{k_y})$. Our definition of a staircase implies that the area $\mathcal{A}_1 = R(v_1, v_2) \cup R(v_2, v_3) \cup \ldots \cup R(v_{n-1}, v_n) \cup R(v_2, c) \cup R(v_3, c) \cup \ldots \cup R(v_{n-1}, c)$ drawn in Figure 3 (a) contains no further points of P besides the staircase points. Assume to the contrary that a point p lies in the area $\mathcal{A}_1$. Then for at least one of the sequence points v_i the x- or y-neighboring point in $Q_3(v_i)$ is p and not as required b^x or b^y. The staircase would split into at least two staircases (see Figure 3 (b)). Also the areas $\mathcal{A}_2 = \{p \in \mathbb{R}^2 \,|\, p_x \leq v_{1_x} \text{ and } v_{n_y} \leq p_y \leq v_{1_y}\}$ and $\mathcal{A}_3 = \{p \in \mathbb{R}^2 \,|\, p_y \leq v_{n_y} \text{ and } v_{1_x} \leq p_x \leq v_{n_x}\}$ contain besides an outer point and possibly a base point no further point of P. If one of the areas $\mathcal{A}_2$ or $\mathcal{A}_3$ contains a point p, then the staircase would split into at least two staircase, too (see Figure 3 (c)).

Now we will prove an important property of Manhattan networks that states that it suffices to compute Manhattan networks of all staircases to get a Manhattan network for the whole instance.

Theorem 4. *Let $P \subseteq \mathbb{R}^2$ be a set of points in the plane and let $\mathcal{S}$ be the set of staircases of P. A network containing Manhattan networks for each staircase of $\mathcal{S}$ is a Manhattan network of P.*

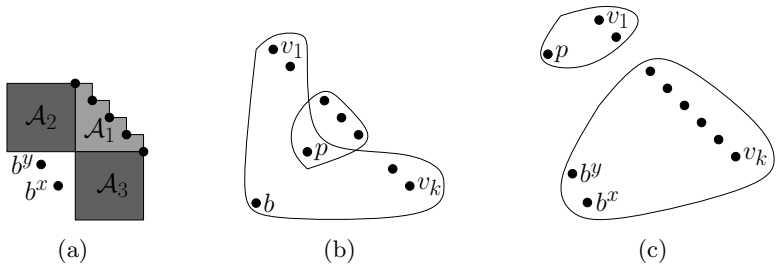


Figure 3: Splitting staircases.

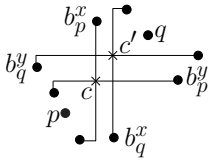


Figure 4: Proof of Theorem 4.

Proof. Let MN be a network containing a Manhattan network for each staircase in $\mathcal{S}$ and let $p, q \in P$ two points. We show, that there exists a shortest path in MN between p and q thus proving that MN is a Manhattan network of P. One can assume that $R(p,q) = \emptyset$ and that p and q do not belong to the same staircase, otherwise the result is immediate from the assumption. Assume $p_x \leq q_x$ and $p_y \leq q_y$. See also Figure 4. Consider the staircase S of the type depicted in Figure 2 (c) with p as a sequence point. Let b_p^x be the x-base point of this staircase and b_p^y the y-base point. Further on, consider the staircase S' of the type depicted in Figure 2 (a) with q as a sequence point. Analogously, let b_q^x be the x-base point of this staircase and b_q^y the y-base point. Now, consider the staircase with base points b_q^x and b_q^y. Since $R(p,q) = \emptyset$, b_p^x lies above and b_p^y on the right of S' and b_q^x below and b_q^y on the left of S. By assumption there are Manhattan paths between the leftmost outer point and b_q^x and the between the bottommost outer point and b_q^y as depicted in Figure 4. By the arrangement of the points in the plane, the two paths intersect in a point c' with $c'_y \geq b^y_{q\,y}$. In the same manner, we get an intersection point c between the Manhattan paths of the outer points and the appropriate base points of the other staircase considered with $c_x \leq b^x_{p\,x}$. Line segments of the (c, b_p^x)- and the (c', b_q^y)-path form a shortest path between the two points c and c' by their location in the plane. There are a shortest (p, c)- and a shortest (p', c')-path in the Manhattan networks for the two staircases by assumption. The two paths form together with the (c, c')-path a shortest (p, q)-path proving the theorem. □

Our algorithm partitions the global Manhattan network problem into disjoint local Manhattan network problems for staircases by inserting line segments which separated the staircases of each other. We construct a boundary for each staircase which is defined as follows:

Definition 5. *A* staircase boundary *for a sequence $(v_1, \ldots, v_k)$ of a staircase with base points b^x and b^y consists of a shortest path for any consecutive sequence points v_i and v_{i+1}, $1 \leq i \leq k-1$, as well as a shortest path between v_1 and b^x and between v_k and b^y.*

The boundary of a staircase is not unique. See Figure 5 for different examples of staircase boundaries for the same point set.

As required later, we want to define a variant of a staircase boundary which we call *extended staircase boundary*. The only difference in the definition for an extended staircase boundary is

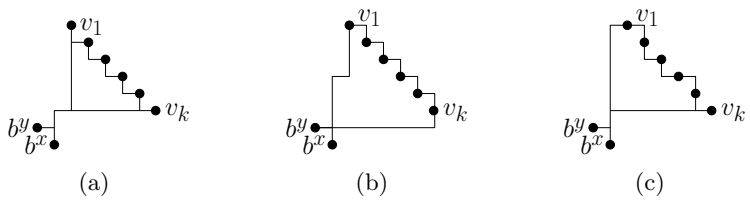


Figure 5: Different staircase boundaries for the same staircase

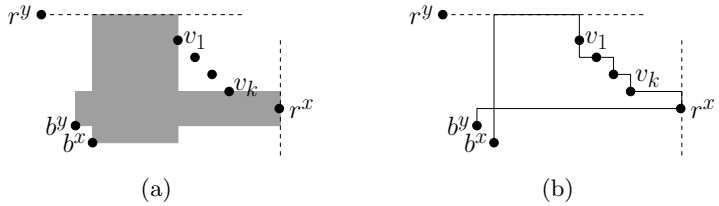


Figure 6: (a) Extended base rectangles. (b) Extended staircase boundary.

that we do not require a shortest path between the outer points and the base points. Nevertheless, there has to be a path lying in a certain area, the so-called *extended base rectangles*.

Definition 6. *Let $(v_1, \ldots, v_k)$ be a staircase sequence with base points b^x and b^y. Let r^y be the y-neighboring point in $Q_2(v_1)$. If there exist no points in $Q_2(v_1)$ then let $r^y = v_1$. Let r^x be the x-neighboring point in $Q_4(v_k)$. If there exist no points in $Q_4(v_k)$ then let $r^x = v_k$. The* extended x-base rectangle *is the rectangle $R((v_{1_x}, r^y_y), b^x)$. The* extended y-base rectangle *is the rectangle $R((r^x_x, v_{k_y}), b^y)$.*

See Figure 6 (a) for an example of the extended base rectangles. Now, we can define the extended staircase boundary.

Definition 7. *An* extended staircase boundary *for a sequence $(v_1, \ldots, v_k)$ of a staircase with base points b^x and b^y consists of a shortest path for any consecutive sequence points v_i and v_{i+1}, $1 \leq i \leq k-1$, as well as a (v_1, b^x)-path inside the extended x-base rectangle and a (v_k, b^y)-path inside the extended y-base rectangle.*

See Figure 6 (b) for an example of an extended staircase boundary.

Since the region surrounded by the staircase boundary plays an important role, we want to name it by the following definition:

Definition 8. *The* (extended) staircase area *of a staircase S with respect to its boundary B is the interior of the region of the plane bounded by B.*

It is essential for our algorithm that after we have assigned an (extended) staircase boundary for each staircase it suffices to compute Manhattan networks for the staircases to achieve a Manhattan network for the complete instance by Theorem 4. The next lemma proves that it suffices to consider for each staircase the interior of the staircase area.

Lemma 9. *Let B be the staircase boundary or extended staircase boundary of a staircase with sequence $(v_1, \ldots, v_k)$ and base points b^x and b^y. The minimum Manhattan network for the staircase given B lies completely inside the staircase area defined by B.*

Proof. To prove the lemma we have to argue that no shortest path in the base rectangles takes course outside the staircase area. Let c be the cut point of the (v_1, b^x)- and the (v_k, b^y)-path.

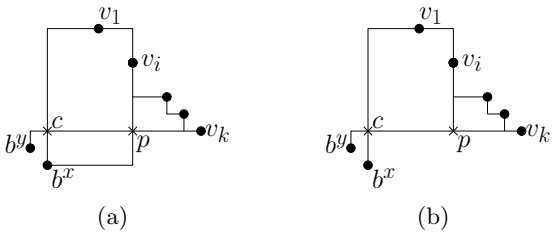


Figure 7: Proof of Lemma 9.

Assume there exists a path from a point v_i, $1 \leq i \leq k$, with parts of the path lying outside the staircase area. See Figure 7. However, this path crosses the boundary segments at a point p, that is the path crosses the boundary segment either between v_1 and b^x or between v_k and b^y. Furthermore this intersection appear before the point c. But then we also have a shortest path to b^x and b^y using the shortest (v_i, p)-, (p, c)-, (c, b^x)- and (c, b^y)-path, respectively. We get a shorter Manhattan network if we delete all segments running outside the staircase area, in contrast to the minimality. □

That is, after we computed an (extended) staircase boundary for each staircase, each Manhattan network for a staircase lies completely inside the staircase area. Obviously, the staircase areas are disjoint regions. Thus, we split the problem into a set of independent subproblems of Manhattan networks for staircases.

3. A 2-Approximation of Minimum Manhattan Networks

Our 2-approximation is composed of two phases. It partitions the global Manhattan network problem into disjoint local Manhattan network problems for staircases by inserting line segments which separate the staircases of each other. This general approach to partition the problem into a set of Manhattan network problems for staircases is used by all combinatorial approaches (see for example [1], [3], [9], [10], [11] or [12]). For this, we compute a set of line segments containing a so-called (extended) staircase boundary for each staircase. Our approach to compute the staircase boundaries is quite similar to that of Nouioua[15]. These staircase boundaries partition our problem into a set of subproblems asking for Manhattan networks for staircases.

It is essential for our algorithm that our definition of staircases delivers a disjoint partition of the plane. More precisely, if we choose for each staircase a staircase boundary then no edge inside a staircase area can contribute to a shortest path between points belonging to other staircases. This characteristic is necessary since on this account we can afterwards compute the Manhattan networks for each staircase independently of the other staircases by Lemma 9. The choice of the staircase boundaries is the crucial point to get an optimal solution for a Manhattan network problem. See Figure 8 (a) and (b) for an instance with two differently chosen staircase boundaries. In this example there are two staircases, first the one with sequence $(p_1, \ldots, p_7)$ and base b and second the one with sequence $(q_1, \ldots, q_5)$ and base p_4. We see that the choice of the boundary for the first staircase affects the staircase area of the second staircase. In Figure 8 (b) we see that the two boundaries share the same edges. If for each staircase the optimal staircase boundary would be known, we could run a dynamic program for each staircase to get a minimum Manhattan network for the complete instance. But, if we want to try all possibilities this leads to an exponential algorithm. To get a polynomial algorithm we have to guess a boundary. Our algorithm computes a boundary for each staircase by considering neighboring points. See Section 4 for a detailed description. The approach to consider neighboring point pairs was also considered by Benkert et al. [1], Chepoi et al. [3], Guo et al. [10, 11] and Kato et al. [12]. Kato et

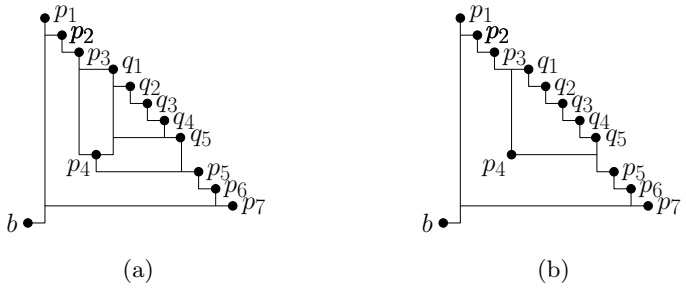


Figure 8: Nested staircases.

al. introduced the notion of a *generating set*, i. e., a set of pairs of points for which it suffices to compute shortest paths to obtain a Manhattan network for the whole instance. Benkert et al. split the generating set into two subsets for which they compute shortest paths separately. To establish shortest paths for all point pairs of the first set (consisting of neighboring point pairs), they altogether use the length of a minimum Manhattan network plus the width and height of the minimum bounding box of P. We use only one time the length of a minimum Manhattan network to get these paths.

To achieve the desired approximation ratio it is important that our staircase areas are at least as small as the staircase areas of a minimum Manhattan network coming from line segments inside rectangles defined by neighboring points. The algorithm we present guarantees that each staircase area $\mathcal{A}_{app}$ computed in Phase I is contained in a staircase area $\mathcal{A}_{opt}$ of boundaries lying inside the neighboring point area $\mathcal{N}$ of a minimum Manhattan network (i. e., $\mathcal{A}_{app} \subseteq \mathcal{A}_{opt}$). With this at hand, we can partition the plane into two regions. To achieve this partition, our approach as well as the approach of Benkert et al. [1] has to add further segments into the first region. Benkert et al. can bound the length of the segments (including the segments satisfying neighboring point pairs) by three times the length of a minimum Manhattan network in this region. The essential improvement of this paper is that we get this result by using only two times the length of a minimum Manhattan network in this region. Afterwards, we use the same approach of Benkert et al. to compute Manhattan networks of staircases. Since they do not explicitely use and prove the above property comparing the staircase areas with the areas of a minimum Manhattan network, we cannot directly applay their algorithm to compute Manhattan networks of staircases. We give a description and prove the approximation ratio in Section 6. To get the overall approximation ratio we compare the length of a minimum Manhattan network in each area separately. This was also done by Benkert et al. In each area we use line segments of length at most twice the length of a minimum Manhattan network inside the appropriate area. Altogether, our network has length at most twice the length of a minimum one.

The algorithm of Guo et al. [11] first computes for neighboring point pairs shortest paths by computing so-called nice vertical and nice horizontal covers and adding so-called switch segments. The nice covers together with the switch segments deliver a staircase boundary for each staircase. Afterwards they use the same strategy to compute 2-approximations for all staircases to get a Manhattan networks for the complete instance. The difference between the algorithm of Guo et al. and our algorithm is that they do not guarantee that each staircase area $\mathcal{A}_{app}$ computed in the first phase is contained in a staircase area $\mathcal{A}_{opt}$ of boundaries lying inside the neighboring point area $\mathcal{N}$ of a minimum Manhattan network (i. e., $\mathcal{A}_{app} \subseteq \mathcal{A}_{opt}$). Without this characteristic, they cannot use the 2-approximation algorithm for staircases to guarantee an overall approximation ratio of 2 because they do not use the optimal boundary for their staircases (minimizing the staircase area). Thus, they must use inside staircase areas longer line

Figure 9: The algorithm of Guo et al. [11].

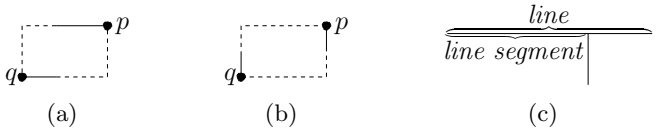


Figure 10: (a) An x-connection between the points p and q. (b) A y-connection between the points p and q. (b) Line and line segment.

segments. The concrete mistake in the approximation analysis lies in the proof of Lemma 5. The set C_2^* is a set of line segments of an optimal Manhattan network. They compare the length of the line segments in C_2^* with the length of the line segments of nice vertical and horizontal covers. In the last paragraph of the proof they examine two points p and q which are considered by the algorithm to compute a nice vertical cover. They define the set $C_2^* \cap ([p, p'] \cup [q, q'])$ and argue that this set contains segments of length at least l. Assume that the set $([p, p'] \cup [q, q'])$ contains the solid line segment as depicted in Figure 9.. The set C_2^* contains the dashed line segment. Then, the set $C_2^* \cap ([p, p'] \cup [q, q'])$ is empty. This differs from the statement that the set contains segments of length at least l. At this point of the analysis Guo et al. have to intersect C_2^* not with segments but with the rectangle $R(p, q')$. As a consequence, in the proof of Theorem 3 they cannot use the statement $(C_V \cup C_H) \cap S_U = \emptyset$ (S_U are the staircase areas without the boundary segments), but have to intersect the rectangles defined by points p and q with the staircase areas. But then, they cannot guarantee that the regions are disjoint and the proof does not work.

4. Computing Boundaries

Our algorithm computes for each staircase a staircase boundary by considering neighboring points.

For two points $p, q \in P$ forming a critical rectangle, a minimum Manhattan network obviously contains line segments of length at least $|p_x - q_x|$ and $|p_y - q_y|$ in the respective dimension inside $R(p, q)$. An *x-connection* for a rectangle $R(p, q)$ is a set of horizontal line segments located inside $R(p, q)$, having the total length $|p_x - q_x|$, and any vertical line passing via a point of $R(p, q)$ intersects at least one of the segments. A *y-connection* for a rectangle $R(p, q)$ is a set of vertical line segments located inside $R(p, q)$, having the total length $|p_y - q_y|$, and any horizontal line passing via a point of $R(p, q)$ intersects at least one of the segments. See also Figure 10 (a) and (b).

By a *line segment* ℓ we denote a horizontal or vertical line which is intersected by a perpendicular line only at the endpoints of ℓ. That is, a line is partitioned into several line segments if a perpendicular line segment ends at the line or crosses the line. See Figure 10 (c).

Kato et al. [12], Benkert et al. [1] and Guo et al. [10, 11] use the notion of an *horizontal cover*. A set of horizontal line segments H is an *horizontal cover* if for any vertical line ℓ and any rectangle $R(p, q)$ of y-neighboring points $p, q \in P$ that ℓ intersects, it holds $\ell \cap R(p, q) \cap H \neq \emptyset$. A *nice horizontal cover* is a horizontal cover H with each line segment of H containing a point of P. A *nice vertical cover* is defined symmetrically. For an illustration of Algorithm 1 NICE HORIZONTAL COVER see again Figure 10 (a) and Figure 11 (a).

Similarly, we get a *nice vertical cover* (see Figure 10 (b) and Figure 11 (b) for illustration).

Algorithm 1 Nice horizontal cover

Require: A set $P \subseteq \mathbb{R}^2$ of points.

1: Let $H = \emptyset$.
2: Sweep over the points of P bottom-up. Let p be the current point and q be the previously processed point (i. e., p is the upper y-neighbor of q).
3: Add to H the portion of $[(q_x, p_y), p]$ that is complementary to $H \cap [q, (p_x, q_y)]$. (See Figure 10 (a).)

It is obvious that the algorithms compute nice covers. To achieve a lower bound on the required length of a nice horizontal and vertical cover we define the area defined by neighboring points.

Definition 10. *The* neighboring point area $\mathcal{N}$ *of a set* $P \subseteq \mathbb{R}^2$ *of points in the plane is the union of all rectangles defined by neighboring points. That is,*

$$\mathcal{N} = \bigcup_{p,q \in P \text{ x- or y-neighboring}} R(p, q).$$

The neighboring point area can also be seen as the union of al strip (rectangles) as defined in [3, 11]. The total length of H and V is required in any Manhattan network.

Lemma 11. *The total length of* $H \cup V$ *is at most the length of a minimum Manhattan network inside the neighboring point area* $\mathcal{N}$*.*

Proof. Consider a nice horizontal cover H. We sweep over the points of P from bottom to top. If for two consecutive points there is not yet an x-connection inserted to the horizontal cover, then we insert it. We know that in any case in the minimum Manhattan network there must be a shortest path and therefore an x-connection between these two points. So we can insert an x-connection to the nice horizontal cover without inserting a segment of length larger than a segment between these points in the minimum Manhattan network. Since the two points are y-neighboring the considered rectangle is inside the neighboring point area $\mathcal{N}$.

The same holds for the nice vertical cover in comparison to the vertical segments of a minimum Manhattan network. □

Our algorithm computes first a nice horizontal and vertical cover H and V. See Algorithm 2 Compute Boundaries for a detailed description. Some of the lines of $H \cup V$ inside critical rectangles do not yet contribute to shortest paths of neighboring points. One end of such a line is neither a point of P nor incident to a perpendicular line. We shift these lines inside the appropriate rectangles to get a connection of the two defining points. See Steps 2 through 7. See Figure 11 for an example of such a covering and moving step. After these steps we have shortest paths between x- or y-neighboring points except for very special configurations which we handle in step 9.

To analyze the algorithm it is essential to keep in mind the following two facts.

Fact 12. *For two* x*-neighboring points* $p, q \in P$ *the set* V *constructed by step 1 of Algorithm 2* Compute Boundaries *contains a* y*-connection between* p *and* q *consisting of at most two lines. For two* y*-neighboring points* $p, q \in P$ *the set* H *constructed by step 1 of Algorithm 2* Compute Boundaries *contains an* x*-connection between* p *and* q *consisting of at most two lines.*

By definition of a nice horizontal cover at least one endpoint of a line segment of H is a point p of P. See Figure 12 for the following fact.

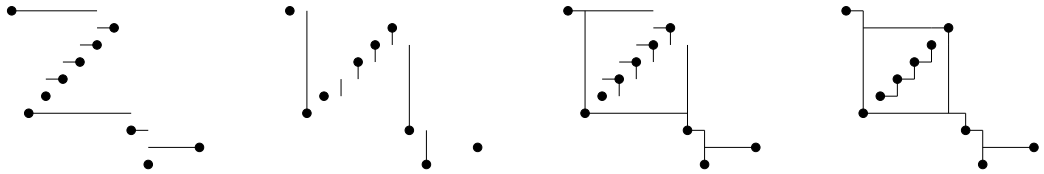


Figure 11: (a) Segments inserted by the nice horizontal cover. (b) Segments inserted by the nice vertical cover. (c) Segments of the nice horizontal and vertical covers together. (d) Network after the moving steps.

Algorithm 2 COMPUTE BOUNDARIES

Require: A set $P \subseteq \mathbb{R}^2$ of points.

Let H be a nice horizontal cover and V be a nice vertical cover.
for each line segment $\ell_h \in H$ considered from top to bottom **do**
    if one endpoint of ℓ_h is neither a point of P nor a point of a line segment of V **then**
        Move ℓ_h downwards until it hits either a point of P or an endpoint of a line segment of V.
for each line segment $\ell_v \in V$ considered from right to left **do**
    if one endpoint of ℓ_v is neither a point of P nor a point of a line segment of H **then**
        Move ℓ_v to the left until it hits either a point of P or an endpoint of a line segment of H.
Set $MN = H \cup V$.
for all x- or y-neighboring points $p, q \in P$ **do**
    if there is no shortest (p, q)-path in MN **then**
        Add to MN the shortest line segment needed to establish a shortest (p, q)-path.
return MN.

Fact 13. *Let $[p, p']$ be a line in H with $p \in P$ and $p' \notin P$. Then p' has an x-coordinate of a point below p' either y-neighboring to p or y-neighboring to a y-neighboring point of p.*

In the same matter, if an endpoint p' of a line segment of V is not in P and the other endpoint is a point $p \in P$ then the point p' has y-coordinate of a point on the left of p either x-neighboring to p or x-neighboring to an x-neighboring point of p.

Our aim is to prove that the algorithm computes for each staircase an (extended) staircase boundary.

Note that for two x-neighboring points p and q we add vertical line segments and for two y-neighboring points we add horizontal line segments. Nevertheless, in the rectangle defined by two x-neighboring points also lies an x-connection and in the rectangle defined by y-neighboring points also lies a y-connection.

Lemma 14. *Let MN be the set of line segments computed by Algorithm 2* COMPUTE BOUNDARIES. *For two x-neighboring points $p, q \in P$ the set MN contains at least one x-connection inside $R(p, q)$. Each x-connection in $R(p, q)$ consists of exactly one segment of length $|p_x - q_x|$. For two y-neighboring points $p, q \in P$ the set MN contains at least one y-connection inside $R(p, q)$. Each y-connection in $R(p, q)$ consists of exactly one segment of length $|p_y - q_y|$.*

Proof. Let $p, q \in P$ be x-neighboring and let $p_x \leq q_x$. There exists at least one x-connection inside $R(p, q)$ because either p and q are also y-neighboring or there exists at least one point r with $p_y < r_y < q_y$ establishing an x-connection.

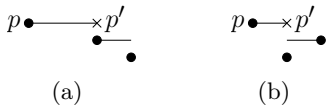


Figure 12: An example for Fact 13.

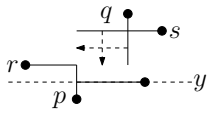


Figure 13: Proof of Lemma 15.

We have to prove that each x-connection in $R(p,q)$ consists of a single segment. Each x-connection is established by two y-neighboring points. Assume there is an x-connection for two y-neighboring points $p', q' \in P$ with $R(p',q')$ intersecting the rectangle $R(p,q)$ and consisting of two line segments inside $R(p,q)$. Then by Fact 13 there exists a point r with $p_x < r_x < q_x$. But then p and q would not be x-neighboring in contradiction to the assumption. Thus, each x-connection inside $R(p,q)$ consists of a single segment. □

Now we prove that if the y-connection for two x-neighboring points $p, q \in P$ in $R(p,q)$ consists of two line segments then p and q are connected by a shortest path after step 7.

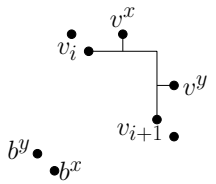
Lemma 15. *Let MN be the set of line segments after step 8 of Algorithm 2* Compute Boundaries. *If for two x-neighboring points $p, q \in P$ the y-connection consists of two line segments or for two y-neighboring points $p, q \in P$ the x-connection consists of two line segments then MN contains a shortest (p,q)-path.*

Proof. W. l. o. g. let $p, q \in P$ be x-neighboring and let $p_x \leq q_x$ and $p_y \leq q_y$. Assume p and q are not connected by a shortest path. Let p' be the other endpoint of the line segment incident to p of the y-connection and q' the endpoint of the line segment incident to q. See Figure 13. By Fact 13 there exists a point r in $Q_2(p)$ with y-coordinate $r_y = p'_y = q'_y$. This point r induces an x-connection inside a rectangle $R(r,s)$ for a point s either being q or a point $s \in Q_4(q)$. By Lemma 14 the x-connection consists inside $R(p,q)$ of exactly one line segment and lies at the height of s. Either we move this line segment down to p' (step 4 of Algorithm 2) or the line segment $[(q_x, s_y), q']$ to the left to p' (step 5 of Algorithm 2) to establish a shortest (p,q)-path. □

With this at hand we now show that we get the desired extended staircase boundaries. First we prove that two neighboring sequence points are connected by a shortest path.

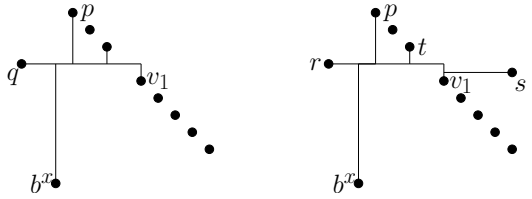
Lemma 16. *Let MN be the set of line segments computed by Algorithm 2* Compute Boundaries. *For each staircase sequence the set of line segments in MN contains a shortest path for consecutive staircase sequence points.*

Proof. Let $(v_1, \ldots, v_k)$ be a staircase sequence. Let v_i and v_{i+1}, $1 \leq i < k$, be two consecutive staircase sequence points with $v_{i_x} \leq v_{i+1_x}$ and $v_{i_y} \geq v_{i+1_y}$ with base points bottom-left (of type as in Figure 2 (a)). If v_i and v_{i+1} are x- or y-neighboring then they are connected by a shortest path (step 11). Thus, assume v_i and v_{i+1} are neither x- nor y-neighboring. Let v^x be the x-neighboring point of v_{i+1} on the left of v_{i+1} and v^y be the y-neighboring point of v_i below v_i. Since v_i and v_{i+1} belong to the same staircase, v^x lies above v_i and v^y on the right of v_{i+1}. See Figure 14. Since v^x is the x-neighboring point of v_{i+1} there is a shortest (v^x, v_{i+1})- and since v^y is the y-neighboring point of v_i there is a shortest (v^y, v_i)-path. Together these paths establish a shortest (v_i, v_{i+1})-path. □



(a)

Figure 14: Proof of Lemma 16.



(a) (b)

Figure 15: Proof of Theorem 17.

With the last two lemmas at hand we can finish our statement that the set MN contains an extended staircase boundary for each staircase.

Theorem 17. *Let MN be the set of line segments computed by Algorithm 2* Compute Boundaries. *For a staircase with base points b^x and b^y the set MN contains paths from b^x and b^y to the outer points of the staircase sequence in the appropriate extended base rectangles.*

Proof. Let $(v_1, \ldots, v_k)$ be a staircase sequence with base points b^x and b^y. If b^x and v_1 are x-neighboring then they are connected by a shortest path after step 11. Thus assume b^x and v_1 are not x-neighboring. Let p be the point x-neighboring to b^x on the right of b^x. See Figure 15. Since p is x-neighboring to b^x we know that there exists a shortest (p, b^x)-path after step 11. Consider the point q y-neighboring to v_1 above v_1. We distinguish two cases. First assume q lies to the left of v_1. Since v_1 is the outer point of the staircase and p is x-neighboring to b^x the point q lies on the left of b^x and below p. Again, there exists a shortest (q, v_1)-path due to the neighborhood of q and v_1. Together with the shortest (p, b^x)-path we get a (v_1, b^x)-path in the extended x-base rectangle. See Figure 15 (a).

Now assume the point q lies to the right of v_1. Since p does not belong to the staircase sequence $(v_1, \ldots, v_k)$ there is a point above v_1 and to the left of b^x. Let r be the one with smallest y-coordinate (i. e., the y-neighboring point in $Q_a(v_1)$). The point r is y-neighboring below to a point s on the right of v_1 (s can be q or a point above q). There exists a shortest (r, s)-path. Since b^x is x-neighboring to v_1 in $Q_3(v_1)$ but is not globally x-neighboring, the point t x-neighboring to v_1 on the left of v_1 lies above r. Again there exists a shortest (v_1, t)-path. Together with the shortest (p, b^x)- and (r, s)-path we get a (v_1, b^x)-path inside the extended x-base rectangle. See Figure 15 (b).

In almost the same manner we can prove that we get a (v_k, b^y)-path inside the extended y-base rectangle. These two paths together establish also a (v_1, b^y)- and a (v_k, b^x)-path closing the proof. □

We can conclude with the following corollary:

Corollary 18. *The set MN of line segments computed by Algorithm 2* Compute Boundaries *contains for each staircase an extended staircase boundary.*

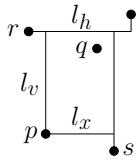


Figure 16: Proof of Theorem 19.

Now we prove that we use at most the length of a minimum Manhattan network inside the neighboring point area.

Theorem 19. *Let MN be the set of line segments computed by Algorithm 2* Compute Boundaries. *The total length of the line segments is at most the part of a minimum Manhattan network located inside the neighboring point area $\mathcal{N}$.*

Proof. By Lemma 11, the nice horizontal and vertial covers H and V have length at most the length of a minimum Manhattan network inside $\mathcal{N}$. Processing steps 2 to 7 does not increase the length of the segments in MN. Now, we will consider the line segments added by step 9. Assume $p, q \in P$ are x-neighboring and let $p_x \leq q_x$ and $p_y \leq q_y$. For the following explanation see Figure 16. Assume after step 7 p and q are not connected by a shortest path. By Lemma 15 the y-connection consists of one segment ℓ_v. The segment ℓ_v is either incident to p or to q, say ℓ_v is incident to p. By Lemma 14 there exists at least one x-connection inside $R(p, q)$ and each x-connection consists of exactly one line segment running completely through $R(p, q)$. The topmost x-connection lies below q (otherwise we have already a shortest (p, q)-path). Furthermore, ℓ_v ends at a horizontal line segment ℓ_h of H (otherwise we could move the part of l_v above the topmost x-connection to establish a shortest (p, q)-path). The presence of the segment ℓ_h indicates that there exists a point $r \in P$ in $Q_2(p) \cap Q_2(q)$. The segments ℓ_h and ℓ_v are part of a shortest (p, r)-path. For this path it is necessary that ℓ_v is incident to p and cannot be moved to the right to the width of q. Now consider an x-connection ℓ_x inside $R(p, q)$. The line segment ℓ_x is caused by a point $s \in P$ in $Q_4(q)$ and is (possibly together with a segment of ℓ_v or other segments) necessary for a shortest (p, s)-path. The segment ℓ_x cannot be moved upwards (to the height of q) to establish a shortest (p, q)-path because we would lose the shortest (p, s)-path. We see that with the given x- and y-connections of the nice horizontal and vertical covers H and V we cannot get simultaneous shortest (p, r)-, (p, s)- and (p, q)-paths (see again Figure 16). Thus, if we add to MN the shortest line segment to achieve a shortest (p, q)-path the length of MN is already at most the length of the line segments of a minimum Manhattan network to connect points being x- or y-neighboring (that is, at most the length of line segments inside $\mathcal{N}$). □

Last, we want to mention the running time of the algorithm.

Lemma 20. *The running time of Algorithm 2* Compute Boundaries *for a set P of n points is $O(n \log n)$.*

Proof. First of all we must sort the points of P. This can be done in time $O(n \log n)$. The sweeps to get nice horizontal and vertical covers can be transformed in $O(n)$ just as the sweeps used by steps 2 through 7. To find all staircases we simply have to assign to each point its two base points by Definition 1. Thus we can find all staircases in time $O(n)$. This yields the total running time for the algorithm of $O(n \log n)$. □

Figure 17: An up-switch.

5. The Algorithm

See Algorithm 3 Smallest Staircases for a detailed description of our algorithm. In a first step, we fix with Algorithm 2 Compute Boundaries the extended boundaries of the staircases. Afterwards, we perform some improvements to the boundaries. For this we want to define an operation called *switch* used by Procedure 4 Make Envelope which moves some line segments of the nice covers. All line segments of a minimum Manhattan network and also all segments picked by our algorithm lie inside the smallest rectangle containing all input points. Furthermore, we can limit the considered area to a smaller region such that we need segments of shorter length to connect points inside this regions. See Figure 17 and 18. For two line segments $\ell_h \in H$ and $\ell_v \in V$ let $R(\ell_h, \ell_v)$ be the smallest rectangle containing ℓ_h and ℓ_v completely. A *corner* is a point $c \notin P$ where two line segments of H and V end. A corner c has to be incident to exact two line segments. Thus, for two line segments $\ell_h \in H$ and $\ell_v \in V$ forming a corner, the segments ℓ_h and ℓ_v both lie on boundary edges of $R(\ell_h, \ell_v)$.

Definition 21. *Let H be a nice horizontal cover and V be a nice vertical cover of a set $P \subseteq \mathbb{R}^2$ of points in the plane. Let $\ell_h \in H$ and $\ell_v \in V$ be two line segments forming a corner. A* switch *of ℓ_h and ℓ_v is an operation of moving the line segments to the opposite sides of the rectangle $R(\ell_h, \ell_v)$. An* up-switch *is performed if ℓ_h lies at the bottom of $R(\ell_h, \ell_v)$ before the switch-operation, otherwise we call it* down-switch.

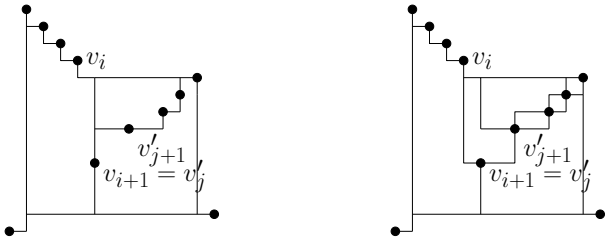
All points connected before a switch-operation by a shortest path are still connected afterwards. The length of a network is not increased by a switch-operation.

With Procedure 4 Make Envelope we get a region contained in the *Pareto envelope*. This region was first considered in the context of Manhattan networks by Chepoi et al. [3] and is called *Pareto envelope*. Given a set $P \subseteq \mathbb{R}^2$ of points in the plane, a point $q \in \mathbb{R}^2$ is called *efficient* point of P [2, 3, 18] if there does not exist another point $r \in \mathbb{R}^2$ such that $d(r, p) \leq d(q, p)$ for each $p \in P$ and $d(r, p') < d(q, p')$ for at least one $p' \in P$. The *Pareto envelope* of P is the set of all efficient points. The Pareto envelope can be computed in time $O(n \log n)$ [2]. The Pareto envelope $\mathcal{P}$ can also be characterized by $\mathcal{P} = \cap_{p \in P} \cup_{q \in P} R(p, q)$. Procedure 4 Make Envelope called in step 2 moves the segments found by Algorithm 2 Compute Boundaries to circumscribe an area as small as possible (i. e., the Pareto envelope) in the following way. The segments computed by Algorithm 2 Compute Boundaries contain for each staircase an extended staircase boundary. Procedure 4 Make Envelope examines all horizontal line segments above or below which no further line segments lie and tries to switch these line segments to diminish the staircase areas. The same is done with the vertical line segments which on the left or on the right no further segments have. (For our approach it is not necessary to prove formally that the area we generate by calling Procedure 4 is inside the Pareto envelope.) See Figure 18 (a) and (b) for an example of the set of line segments in MN before and after calling Procedure 4 Make Envelope.

Now, we have chosen an extended staircase boundary for each staircase. Recall that the only remaining connections to get a Manhattan network are paths inside staircase areas. The length of the chosen boundaries is at most the length of a minimum Manhattan network by Theorem 19. Unfortunately, our choice maybe not optimal. That is, we possibly get larger staircase areas as the staircase areas of a minimum Manhattan network. The Manhattan network in the interior

(a) (b)

Figure 18: (a) The set MN before calling Procedure 4 MAKE ENVELOPE. (b) The set MN after calling Procedure 4 MAKE ENVELOPE.



(a) (b)

Figure 19: (a) The set MN before calling Procedure 5 MAKE SMALLEST SEQUENCE. (b) The set MN after calling Procedure 5 MAKE SMALLEST SEQUENCE.

of the staircase area has length exceeding the length of a minimum Manhattan network in the interior of the staircase area of the optimal boundary. It may well pay off to use more boundary segments in order to decrease the staircase areas. See Figure 8 for an example. If we diminish the staircase areas, the lengths of the networks inside the staircase areas will decrease as well. Note that until now we only examined x- or y-neighboring points (i. e., we only added segments inside the neighboring point area $\mathcal{N}$). We do not know the optimal staircase boundaries, nevertheless we now want to make the staircase areas as small as possible with line segments inside $\mathcal{N}$. On this behalf we examine the staircases and their boundaries.

We differentiate whether the considered line segments are segments between neighboring sequence points or between the base points and the outer points. The first segments are examined by Procedure 5 MAKE SMALLEST SEQUENCE called in step 3 of Algorithm 3. Assume that two neighboring staircase sequence points v_i and v_{i+1} are x-neighboring. We do not know which shortest path between neighboring sequence points is chosen by a minimum Manhattan network. For this reason we insert the missing line segments of $\partial R(v_i, v_{i+1})$ if they diminish the staircase area (step 2 of Procedure 5). See Figure 19. We have to take care that the length of the selected line segments is not to large. Therefore, we keep in mind the already considered and inserted segments (step 5 and 8 of Procedure 5) and delete superfluous line segments (step 4 and 7 of Procedure 5). In the example in Figure 19, we insert for both, the rectangle $R(v_i, v_{i+1})$ and the rectangle $R(v'_j, v'_{j+1})$ the vertical boundary segments. Assume the algorithm considers first the rectangle $R(v_i, v_{i+1})$. The algorithm marks the two vertical line segments of $R(v_i, v_{i+1})$. If the algorithm considers the rectangle $R(v'_j, v'_{j+1})$ it inserts the remaining vertical boundary segments of it and deletes the vertical segment incident to $v_{i+1} = v'_j$ inside $R(v'_j, v'_{j+1})$ (step 4 of Procedure 5). If we would not delete this segment we could not guarantee that we add at most twice the length of line segments of a minimum Manhattan network inside $R(v_i, v_{i+1}) \cup R(v'_j, v'_{j+1})$. We will prove in Lemma 26 that we preserve shortest paths by this transformation.

Now we consider boundary segments between the base and the outer points. Look at Figure 20 for the following. The vertical line segment incident to b^x takes account for the staircase areas with staircase sequences $(v_1, \ldots, v_4)$ and $(v'_1, \ldots, v'_4)$ being not smallest with respect to the neighboring point area. We look at the x-neighboring point v^x of b^x and add the vertical line

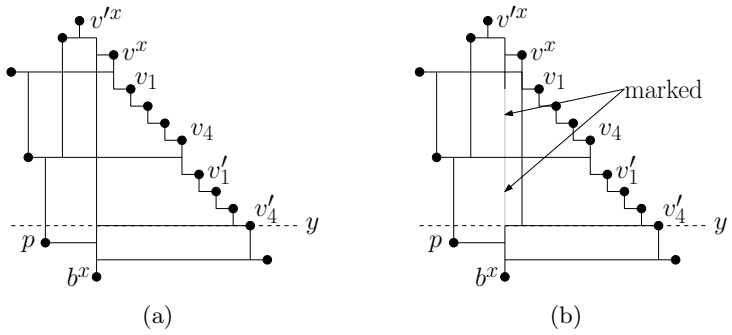


Figure 20: (a) The set MN before calling Procedure 6 Make Smallest Base. (b) The set MN after calling Procedure 6 Make Smallest Base.

segment beginning at v^x which ends at the next horizontal line segment below or at the same height as v'_4. See Figure 20 (b). By Procedure 6 Make Smallest Base which is called in step 4 of the algorithm this is done. Procedure 6 considers all x-base points and examine whether there exist staircases for which the boundary is not optimal according to segments inside rectangles of neighboring points. As above we have to take care about superfluous line segments. The procedure marks all segments which are doubled (step 9 of Procedure 6). See Figure 20 for the marked segments. If in a later step a marked segment is again doubled (to the other side as before), it will be deleted (step 8 of Procedure 6). After calling Procedure 6, for all staircases the boundary between base points and outer points lie in such a way that the staircase area is smallest possible. Procedure 6 describes the proceeding only for staircase sequences lying on the right and above the base point. The same has to be done with the sequences on the left. In an analogous way we have to consider also the y-base points.

The benefit of Procedure 5 Make Smallest Sequence and Procedure 6 Make Smallest Base is, if we allow only line segments in the region defined by rectangles of x- or y-neighboring points (the neighboring point area $\mathcal{N}$), then for each staircase the staircase area is smallest. More precisely, each staircase area $\mathcal{A}_{app}$ computed in Phase I is contained in a staircase area $\mathcal{A}_{opt}$ defined by a boundary inside the neighboring point area $\mathcal{N}$ of a minimum Manhattan network (i. e., $\mathcal{A}_{app} \subseteq \mathcal{A}_{opt}$). This is an important point to guarantee our approximation ratio. Furthermore, we will show that after calling the Procedure 6 Make Smallest Base we inserted at most twice the length of all line segments of a minimum Manhattan network inside the neighboring point area $\mathcal{N}$. In the second phase we compute a 2-approximation for each staircase given the staircase boundary. Since we know that no staircase area of a minimum Manhattan network defined by shortest paths between x- or y-neighboring points is smaller than our staircase areas, we also add for all staircases together at most twice the length of segments of a minimum Manhattan network in the staircase areas.

In Phase II of the algorithm we compute a Manhattan network for each staircase given the staircase boundary with Algorithm 7 Recursion For Staircases achieving for each staircase a Manhattan network with length at most twice the length of a minimum Manhattan network for this staircase. Altogether, Algorithm 3 computes a Manhattan network for a set P of points.

Since the algorithm computes a Manhattan network for each staircase, by Theorem 4 we get the following.

Theorem 22. *Algorithm 3* Smallest Staircases *computes a Manhattan network for the input points P.*

Algorithm 3 SMALLEST STAIRCASES

Require: A set $P \subseteq \mathbb{R}^2$ of points.

Phase I:

Let $MN = H \cup V$ be the output of Algorithm 2 COMPUTE BOUNDARIES.
Let $MN =$ MAKE ENVELOPE(P, H, V).
Let $MN =$ MAKE SMALLEST SEQUENCE(P, MN).
Let $MN =$ MAKE SMALLEST BASE(P, MN).

Phase II:

for each staircase S **do**
    Compute with Algorithm 7 a Manhattan network MN' of S with the staircase boundary computed in Phase I.
    $MN = MN \cup MN'$.
return MN.

Procedure 4 MAKE ENVELOPE

Require: A set $P \subseteq \mathbb{R}^2$ of points and a nice horizontal cover H and a nice vertical cover V of P.

for each horizontal line segment $\ell_h \in H$ under which there do not lie any other line segments of H **do**
    if l_h forms a corner with a vertical line segment $\ell_v \in V$ **then**
        Perform an up-switch if possible.
for each horizontal line segment $\ell_h \in H$ above which there do not lie any other line segments of H **do**
    if ℓ_h forms a corner with a vertical line segment $\ell_v \in H$ **then**
        Perform a down-switch if possible.
return H and V.

6. A 2-Approximation of Staircases

In this section we deal with the problem to compute Manhattan networks for staircases given the staircase boundary. The algorithm recursively partitions a staircase into two new staircases. This proceeding is also known as *thickest-first* partitioning and was analyzed to yield a 2-approximation by Gudmundsson et al. [9] and Benkert et al. [1]. Gudmundsson et al. [9] prove that given a staircase boundary a rectangulation of the polygon defined by the boundary with minimum total edge length is a minimum Manhattan network for the points defining the staircase boundary. Lingas et al. [16] show that a minimum rectangulation of a rectilinear polygon can be computed in time $O(n^4)$. They state, that for a special case of so-called *histograms* a rectangulation can be computed in $O(n^3)$. They introduced a 2-approximation for rectangulations with running time $O(n \log n)$. We make other demands on the staircase boundary and thus we cannot use the algorithm of Gudmundsson et al. [9] directly. Furthermore, we achieve the result by computing a Manhattan network directly without the detour to the rectangulation. The algorithm of Benkert et al. [1] to compute Manhattan networks for staircase boundaries has more analogies to our than the the one of Gudmondsson et al. [9]. For the purpose of self containment and since both the definitions of staircases and the boundary of staircases are not standardized we specify the algorithm adapted to our conditions.

Given the staircase boundary, for a point v_i of the staircase sequence we denote by x_i the missing line segment of a shortest path to the vertical left boundary edge, by y_i we denote the missing line segment of a shortest path to the horizontal bottom boundary edge. See Figure 21

Procedure 5 MAKE SMALLEST SEQUENCE

Require: A set $P \subseteq \mathbb{R}^2$ of points and a set MN of vertical and horizontal line segments.

for each rectangle $R(v_i, v_{i+1})$ defined by neighboring staircase sequence points v_i, v_{i+1} of a staircase with more than two sequence points where v_i and v_{i+1} are at least either x- or y-neighboring **do**
    Insert into MN the missing line segments of $\partial R(v_i, v_{i+1})$ which induce a smaller staircase area.
    if v_i and v_{i+1} are x-neighboring and vertical line segments are inserted by step 2 **then**
        Delete all marked vertical line segments of $\partial R(v_i, v_{i+1})$.
        Mark the remaining vertical line segments of $\partial R(v_i, v_{i+1})$.
    if v_i and v_{i+1} are y-neighboring and horizontal line segments are inserted by step 2 **then**
        Delete all marked horizontal line segments of $\partial R(v_i, v_{i+1})$.
        Mark the remaining horizontal line segments of $\partial R(v_i, v_{i+1})$.
return MN.

Procedure 6 MAKE SMALLEST BASE

Require: A set $P \subseteq \mathbb{R}^2$ of points and a set MN of vertical and horizontal line segments.

for each x-base point b^x **do**
    Let v^x be the x-neighboring point of b^x on its right.
    if there exists a staircase sequence with more than 2 points and b^x as x-base point **then**
        Let $(v_1, \ldots, v_k), k \geq 3$, be the staircase sequence with v_k has the smallest y-coordinate among all staircase sequences with b^x as x-base point.
        Let y be the y-coordinate of the next horizontal line segment of MN below v_k touching the x-coordinate v^x.
        if v^x is not incident to a vertical line segment covering $[v^x, (v^x_x, y)]$ **then**
            Add to MN the line segment $[v^x, (v^x_x, y)]$.
            Delete all marked line segments of $[(b^x_x, v^x_y), (b^x_x, y)]$.
            Mark the remaining line segments of $[(b^x_x, v^x_y), (b^x_x, y)]$.
return MN.

for an illustration. Note, that we do not count the length of the possibly used boundary edges between v_i and the left and bottom boundary, respectively. Let p^x_i be the intersection of x_i with the vertical left boundary edge and p^y_i the intersection of y_i with the horizontal bottom boundary edge.

Our algorithm partitions the staircase into two staircases for which networks are computed recursively. In the partitioning step two new edges are inserted, each of them being a new boundary edge of one of the two new staircases. We get as input an (extended) staircase boundary of a staircase with sequence $(v_1, \ldots, v_n)$ and base points b^x and b^y. We consider the segments x_i and y_i, $1 \leq i \leq n$ to connect a point v_i to the left and bottom boundary segment, respectively. We select the point v_i, $1 \leq i < n$, for which $|x_i| \leq |y_i|$ and $|x_{i+1}| \geq |y_{i+1}|$ holds and add the segments x_i and y_{i+1} to our network and compute recursively Manhattan networks for the staircases with sequence $(v_1, \ldots, v_i)$ and the one with sequence $(v_{i+1}, \ldots, v_n)$. As illustrated in Figure 21 we insert edges x_6 and y_7 and get two new staircases with sequences $(v_1, \ldots, v_6)$ and $(v_7, \ldots, v_{12})$. See Algorithm 7 RECURSION FOR STAIRCASES for a detailed description. For this, assume the intersection point c between the (b^x, v_1)- and (b^y, v_n)-path lies on the origin.

Let $(v_1, \ldots, v_n)$ be a staircase sequence with base points b^x and b^y. Let $\mathcal{A}_{app}$ be the staircase area of the staircase boundary given to Algorithm 7 RECURSION FOR STAIRCASES for the

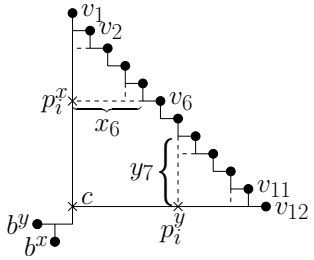


Figure 21: The definition of x_i and y_i.

Algorithm 7 RECURSION FOR STAIRCASES

Require: A staircase sequence $(v_1, \ldots, v_n)$, $n \geq 3$, with boundary.

1: Set $SC = \emptyset$.
2: Let v_i, v_{i+1} be the unique pair of neighbors with $|x_i| \leq |y_i|$ and $|x_{i+1}| \geq |y_{i+1}|$.
3: Set $SC = SC \cup \{x_i\} \cup \{y_{i+1}\}$.
4: Let SC' be the staircase recursively computed for the staircase sequence $(v_1, \ldots, v_i)$.
5: Let SC'' be the staircase recursively computed for the staircase sequence $(v_{i+1}, \ldots, v_n)$.
6: **return** $SC \cup SC' \cup SC''$

instance. A minimum Manhattan network contains shortest (b^x, v_1)- and (b^y, v_n)-paths and shortest (v_i, v_{i+1})-paths, $1 \leq i < n$, inside the neighbored point area $\mathcal{N}$ constituting a staircase boundary B_{opt} inside $\mathcal{N}$. Let $\mathcal{A}_{opt}$ the staircase area of B_{opt}.

Theorem 23. *Algorithm 7* RECURSION FOR STAIRCASES *computes a Manhattan network for a staircase with total length at most* 2 *times the length of a minimum Manhattan network for it if* $\mathcal{A}_{app} \subseteq \mathcal{A}_{opt}$ *holds.*

Proof. Due to the fact that we recusively call the algorithm and in each call, we connect two points of the sequence to the base, we get a Manhattan network for the staircase.

We prove the ratio between the length of the solution of Algorithm 7 and the length of a minimum Manhattan network to be two by the use of an inductive argument over the number of sequence points. Assume we insert in the i-th step a segment x_k and a segment y_{k+1}. Let x_i^{opt} and y_i^{opt}, $1 \leq i \leq n$, the segments x_i and y_i for B_{opt}. We get $|x_i| \leq |x_i^{opt}|$ and $|y_i| \leq |y_i^{opt}|$ because $\mathcal{A}_{app} \subseteq \mathcal{A}_{opt}$ holds. If we insert x_i then it holds $|x_i| \leq |y_i|$ and therefore $|x_i| \leq min\{|x_i^{opt}|, |y_i^{opt}|\}$. In an analogous manner it holds $|y_{i+1}| \leq min\{|x_{i+1}^{opt}|, |y_{i+1}^{opt}|\}$ if we insert y_{i+1}. To connect v_i and v_{i+1} to the cross point a minimum Manhattan network needs at least the length of $min\{|x_i^{opt}|+|y_{i+1}^{opt}|, |y_i^{opt}|, |x_{i+1}^{opt}|\}$. If the minimum is adopted for $|x_i^{opt}|+|y_{i+1}^{opt}|$ then our algorithm takes the right choice. Otherwise we get the following estimation:

$$\begin{aligned} |x_i| + |y_{i+1}| &\leq min\{|x_i^{opt}|, |y_i^{opt}|\} + min\{|x_{i+1}^{opt}|, |y_{i+1}^{opt}|\} \\ &\leq min\{|x_{i+1}^{opt}|, |y_i^{opt}|\} + min\{|x_{i+1}^{opt}|, |y_i^{opt}|\} \\ &\leq 2\, min\{|y_i^{opt}|, |x_{i+1}^{opt}|\} \end{aligned}$$

. So in step k we insert at most two times the required length in the minimum Manhattan network to connect v_i to the base. Furthermore, we split the staircase in two disjoint staircase with smaller number of sequence points. According to the induction hypotheses for these two staircases the approximation ratio holds. □

Theorem 24. *The running time of the* RECURSION FOR STAIRCASES *algorithm is* $O(n \log n)$.

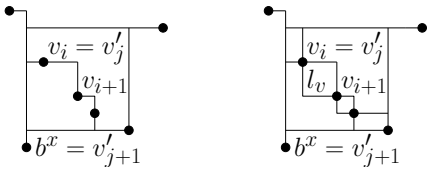


Figure 22: Proof of Lemma 25: The segments before and after calling Procedure 5

Proof. The only time consuming work to be done is step 2 which can be executed by binary seach in time $O(\log n)$. The recursion has to be proceeded $O(n)$ times. Together we get the running time of the algorithm to be $O(n \log n)$. □

7. Analysis of the Algorithm

First we prove that no shortest path is deleted by calling Procedure 5 Make Smallest Sequence. For this consider the following lemma which can be easily seen by considering the possible intersections of sequence rectangles.

Lemma 25. *If a line segment is deleted in step 4 or 7 of Procedure 5* Make Smallest Sequence*, it was already in MN before calling Procedure 5.*

Proof. Let $R(v_i, v_{i+1})$ be a sequence rectangle which is considered by Procedure 5. Let v_i and v_{i+1} be x-neighboring with $v_{i_x} \leq v_{i+1_x}$ and base points b^x and b^y lying bottom-left. See Figure 22. Procedure 5 inserts in step 2 line segments inducing a smaller staircase area. If in step 4 such a segment ℓ_v incident to v_i would be deleted, it has to be contained in a further sequence rectangle $R(v'_j, v'_{j+1})$ of two x-neighboring points v'_j and v'_{j+1}. One of the two points v'_j and v'_{j+1}, say v'_j, has to be the point v_i. Since $R(v_i, v_{i+1})$ and $R(v'_j, v'_{j+1})$ have the segment ℓ_v in common, v_i is an outer point and $b^x = v'_{j+1}$. Before calling Procedure 5 the segment ℓ_v is not in MN. Thus, the complete left boundary of $R(v'_j, v'_{j+1})$ is in MN. Therefore, if Procedure 5 considers the rectangle $R(v'_j, v'_{j+1})$, the segment would not be doubled to the left and therefore not deleted in step 4. □

Now we prove that calling Procedure 5 Make Smallest Sequence does not delete any shortest path.

Lemma 26. *Let $p, q \in P$ connected by a shortest path before calling Procedure 5* Make Smallest Sequence *in step 3 of Algorithm 3* Smallest Staircases*. Then, p and q are connected by a shortest path after step 3.*

Proof. Consider steps 2, 4 and 7 of Procedure 5 Make Smallest Sequence. Let ℓ_v be a vertical line segment that is deleted in step 4. If a line segment is deleted and therefore marked before, this implies that it is also part of a rectangle considered before. The algorithm marks vertical line segments if they are induced by x-neighboring points. Thus, ℓ_v is the intersection of two sequence rectangles $R(p, q)$ and $R(p, r)$ with common point p. That is, the segment ℓ_v is incident to the point p which is x-neighboring to the points q and r either both above or both below p. Assume $r_y \geq q_y \geq p_y$. See Figure 23 (a).

The points p and q and the points p and r are neighboring staircase sequence points of two different staircases. The segment ℓ_v is deleted if it would be doubled to the left and to the right. This implies that p and q belong to a staircase with base points bottom-left and p and r to one with base points bottom-right. See Figure 23 (b). Thus, p and q are also y-neighboring and the

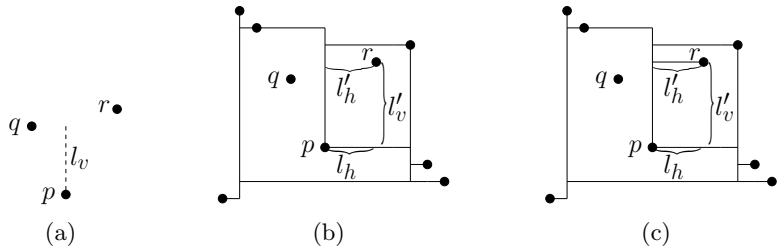


Figure 23: Proof of Lemma 26.

y-connection inside $R(p,q)$ consists of exactly one line segment by Lemma 14. The segment ℓ_v is deleted if the line segments $[q,(q_x,p_y)]$ and $\ell'_v = [r,(r_x,p_y)]$ are inserted by step 2 and the segment ℓ_v is already in MN before calling Procedure 5. Since p and q have its base points bottom-left and p and r bottom right, by step 2 also the horizontal line segments $[p,(q_x,p_y)]$ and $\ell_h = [p,(r_x,p_y)]$ are inserted if they do not already exist. Since p and r are x-neighboring there is a shortest (p,r)-path in MN after step 1 of Algorithm 3. Assume one of the paths between p and q and between p and r, say the (p,r)-path, is missing after calling Procedure 5. That is, either the vertical line segment ℓ'_v or the horizontal line segment ℓ_h, say ℓ_h, is also deleted. By Lemma 25, ℓ_h is determined by a nice horizontal cover in step 1 of Algorithm 3 and belongs to H. Thus, before calling Procedure 5 in MN are the segments $[p,(p_x,r_y)]$, ℓ_h and either $\ell'_h = [r,(p_x,r_y)]$ or ℓ'_v. If ℓ'_h is already in MN after step 1, ℓ_h would not be deleted by step 7 of Procedure 5. See Figure 23 (c). If l'_v is already in MN after step 1, ℓ_v would not be deleted by step 4 of Procedure 5. Therefore, there exists a shortest (p,r)-path after calling Procedure 5. □

Now we examine Procedure 6 Make Smallest Base and motivate why no shortest path is destroyed by calling it. First we show that the procedure is applied only for a very special configuration.

Lemma 27. *The if-clause in step 6 of Procedure 6* Make Smallest Base *is fulfilled only if the x-neighboring points of b^x lie both above or both below b^x.*

Proof. Assume v^x is not incident to a line segment covering $[v^x,(v^x_x,y)]$ (y as defined by step 5 of Procedure 6). Assume $v^x_y \geq b^x_y$. Since b^x and v^x are x-neighboring, b^x is incident to a vertical line segment pointing upwards. This line segment is inserted into MN and could be moved or switched to the width of v^x if there would not be an x-neighboring point of b^x on the other side of b^x prohibiting this. Therefore, this point lies also above b^x. □

Now we can prove that Procedure 6 does not delete any shortest path between neighboring points.

Lemma 28. *All points of P connected by a shortest path before calling Procedure 6* Make Smallest Base *in step 4 of Algorithm 3* Smallest Staircases *are still connected after step 4.*

Proof. Consider steps 8 and 9 of Procedure 6 Make Smallest Base. Assume v^x lies above b^x. If a line segment is marked this implies that it is also considered either by Procedure 5 Make Smallest Sequence (i. e., is part of a rectangle defined by neighboring sequence points) or part of another base rectangle already considered before by Procedure 6.

Assume a shortest path is destroyed. This path can be only be a path between b^x and a point either in the first or second quadrant of b^x. The horizontal line segment with y-coordinate y as defined in step 5 of Procedure 6 which touches the x-coordinate of v^x is justified by two

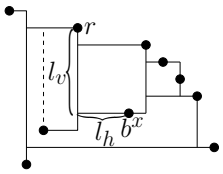


Figure 24: Proof of Lemma 28.

y-neighboring points $p, q \in P$ above or on the same height as b^x. One of the two points, say p, lies to the left of b^x and the other to the right (p can be b^x). Before calling Procedure 6 there exists a shortest (b^x, v^x)-path and a shortest (p, q)-path. Since we add in step 7 the missing line segments of $[v^x, (v^x_x, y)]$ together with the shortest (p, q)-path and the parts of the former (b^x, v^x)-paths with y-coordinates smaller or equal y we already keep after step 8 all shortest paths to points in the first quadrant of b^x. See Figure 20 for an example.

Now assume we lose a shortest path to a point in the second quadrant of b^x because of the absence of a line segment $[(b^x_x, y_1), (b^x_x, y_2)], y_1 \leq y_2$. The deleted segments of $[(b^x_x, v^x_y), (b^x_x, y)]$ are marked during considering the other point $r \in P$ x-neighboring to b^x which lies above b^x by Lemma 27. See Figure 24. If the segment is marked in a preceding step of Procedure 6 Make Smallest Base, by the same argument as above we conserve also all shortest paths to points in the second quadrant of b^x.

Thus, assume b^x and r are neighboring points of a staircase sequence and $[(b^x_x, y_1), (b^x_x, y_2)]$ is marked by step 5 of Procedure 5 Make Smallest Sequence. The appropriate staircase has its base points bottom-left. The line segments $\ell_v = [r, (r_x, b^x_y)]$ and $\ell_h = [b^x, (r_x, b^x_y)]$ are inserted by step 2 of Procedure 5 if they are not already contained in MN.

The line segment ℓ_v is deleted if it is doubled to the left (dashed line segment in Figure 24). That is, ℓ_v is in the set V by Lemma 25. But then the line segment $[(b^x_x, y_1), (b^x_x, y_2)]$ would not be doubled to the left and therefore not marked when considering the sequence rectangle $R(b^x, r)$. See again Figure 24.

Now consider the segment ℓ_h. Since b^x and r are not y-neighboring (otherwise r would lie below the y-coordinate y), the line segment ℓ_h is not marked by step 8 of Procedure 5 Make Smallest Sequence and therefore not deleted when considering $R(b^x, r)$. The line segment ℓ_h is deleted only if it is doubled upwards. A doubling upwards is performed if there is a staircase with more than two sequence points and base points top-right. The point b^x has to be one sequence point. But then the left x-neighbor r of b^x lies not above the height y and we keep a shortest path to r if we delete vertical segments above y. Thus, we do not delete a shortest path to a point in $Q_2(b^x)$.

Altogether we see that calling Procedure 6 Make Smallest Base does not delete any shortest path. □

The next lemma is important to prove the approximation ratio by partitioning the plane in two disjoint regions. Before we state the lemma, we need a further definition.

Definition 29. *The* minimum staircase boundary *of a staircase S is a staircase boundary B inside the neighboring point area $\mathcal{N}$ with a staircase area of minimum size.*

Lemma 30. *After calling Procedure 5* Make Smallest Sequence *and Procedure 6* Make Smallest Base *all staircase boundaries are smallest staircase boundaries.*

Proof. First, consider Make Smallest Sequence. By Lemma 25 the segments inserted by step 2 are not deleted afterwards. That is, after calling 4, for each sequence rectangle the staircase boundary lies in such a way that the staircase area is minimized. Now, we examine

MAKE SMALLEST BASE. After inserting the line segment $[v^x, (v^x_x, y)]$ in step 7 the staircases in $Q_4(b^x)$ with x-base point b^x (of type as depicted in Figure 2 (a)) have a minimum staircase area. The line segment $[v^x, (v^x_x, y)]$ is inserted by considering the x-neighboring points b^x and v^x. Assume a part of the line segment $[v^x, (v^x_x, y)]$ accounting to such a minimum staircase area is deleted afterwards. Before the segment would be deleted, it would be marked. In the following we will show, that a line segment cannot be added, marked and deleted because such a line segment cannot be considered three times by Procedure MAKE SMALLEST BASE. The main argument is that line segments are added, marked and deleted by considering x-neighboring points and that a point at which a line segment is incident to, is x-neighboring to at most two points. Consider the line segment that is deleted afterwards. This segment is first marked and afterwards deleted. Then, the point v^x is a base point of two staircases S and S' with sequence point either in $Q_3(v^x)$ or $Q_4(v^x)$ of type as in Figure 2 (c) or (d), respectively. The staircases S and S' lie below the deleted line segment (otherwise the line segment would not be deleted). Since v^x is the x-base point of S and S' and thus x-neighboring to one sequence point of both staircases, S and S' has to lie both in $Q_3(v^x)$. But v^x can be x-neighboring to at most one point to the left, contradicting that the segment is considered three times by Procedure MAKE SMALLEST BASE. Thus, no segment inserted by step 7 is deleted afterwards, proving the lemma. □

To achieve the approximation ratio our algorithm is allowed to use at most twice the length of a minimum Manhattan network. By Theorem 19 the length of the segments inserted by step 1 is at most the length of a minimum Manhattan network inside the neighboring point area $\mathcal{N}$. As mentioned earlier the steps performed by Procedure 4 MAKE ENVELOPE does not increase the length of the line segments.

Theorem 31. *The total length of the line segments in MN after Phase I of Algorithm 3* SMALLEST STAIRCASES *is at most twice the length of a minimum Manhattan network inside the neighboring point area $\mathcal{N}$.*

Proof. Consider Procedure 5 MAKE SMALLEST SEQUENCE. Assume the actually considered consecutive staircase sequence points v_i and v_{i+1} are x-neighboring. The nice vertical cover contains a y-connection inside the vertical segments of $\partial R(v_i, v_{i+1})$. Each time we insert a vertical line segment by step 2 of Procedure 5, we mark all vertical line segments of $\partial R(v_i, v_{i+1})$. That is, we keep in mind that we doubled the nice vertical cover in $\partial R(v_i, v_{i+1})$. If the algorithm wants to double such a segment a second time for a rectangle $R(v'_j, v'_{j+1})$ (having exactly this line segment of $R(v_i, v_{i+1})$ in common), then this segment will be deleted by step 4. By Lemma 25 no segment inserted by Procedure 5 is itself doubled in a later step of Procedure 5. See Figure 19. Thus, after step 8 the vertical segments are only doubled and not tripled.

Now, consider the horizontal segment inserted by step 2. If v_i and v_{i+1} are not y-neighboring we cannot consult a horizontal line segment of the nice horizontal cover to legitimate it. For this, we use the Pareto envelope. In step 2 of Algorithm 3 we moved the line segments of the nice horizontal and vertical covers to get a smallest area which has to be considered afterwards. No line segment of the Pareto envelope has to be doubled by step 2 because all segments of staircase boundaries belonging to the envelope are lying in such a way that the staircase area is smallest possible. Thus, if the algorithm a horizontal line segment ℓ_h inserts in step 2 we want to justify it by line segments of the Pareto envelope with the same x-coordinates. We have to argue that each line segment of the envelope is used at most once. Assume that a part of two line segments ℓ_h and ℓ'_h are justified by the same segment of the envelope. For this, let v_i and v_{i+1} and v'_j and v'_{j+1} be the two pairs of neighboring staircase sequence points which are both not y-neighboring and for which ℓ_h and ℓ'_h are inserted by step 2. Assume $v_{i_x} \leq v'_{j_x}$. In order

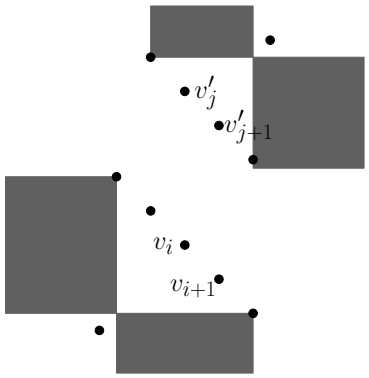


Figure 25: Proof of Theorem 31.

that one segment of the envelope is used to justify ℓ_h and ℓ'_h, it holds $v'_{j_x} < v_{i+1_x}$. Thus, v'_j and v'_{j+1} could be only x-neighboring (as desired by the condition of step 1 of Procedure 5) if $v_{i_x} = v'_{j_x}$ and $v_{i+1_x} = v'_{j+1_x}$. Again by the condition of step 1 of Procedure 5 both sequences have more than two sequence points and since the shaded areas as depicted in Figure 25 are empty as mentioned in Section 2, the lower staircase has its base points below the sequence points and the upper staircase has them above. One of the two horizontal line segments ℓ_h and ℓ'_h is justified by an envelope segment lying below and the other by the upper envelope segment. Again since the shaded areas are empty there cannot lie above or below these two staircases further staircases for whom also horizontal line segments which are added by the same envelope segment. Thus, each envelope segment is used to justify only one segment inserted by step 2.

Now we consider Procedure 6 Make Smallest Base. See again Figure 20. If the if-clause in step 6 is fulfilled, by Lemma 27 b^x is x-neighboring to two points v^x and v'^x above or below it. There is a y-connection in the rectangle $R(v^x, b^x)$ and one in the rectangle $R(v'^x, b^x)$. If v^x is not incident to a vertical line segment covering $[v^x, (v^x_x, y)]$ (y defined as in step 5 of Make Smallest Base) the y-connection is incident to b^x. The algorithm inserts the missing line segments of $[v^x, (v^x_x, y)]$. If segments of $[(b^x_x, v^x_y), (b^x_x, y))]$ are already marked, this implies that also in the rectangle $R(v'^x, b^x)$ there are line segments inserted at the width of v'^x. That is, the line segments of $[(b^x_x, v^x_y), (b^x_x, y))]$ given by the nice vertical cover is doubled already. If the algorithm wants to double it a second time we delete this segment by step 8. After step 8, MN contains at most twice the length of $[(b^x_x, v^x_y), (b^x_x, y))]$ returned by step 1 of Algorithm 3. By the construction of the doubling of Make Smallest Base, for each staircase the doubling is performed to diminish the staircase area. That is, after doubling a segment, on one side of this segment lies the smallest staircase area and the doubled segment could not be doubled into this direction. Thus, no doubled line segment will be itself be doubled. If a segment of H or V is doubled twice, it will be deleted. Thus after Phase I the total length of the line segments in MN is at most twice the length of a minimum Manhattan network inside $\mathcal{N}$. □

We are ready now to prove that our algorithm computes a Manhattan network with approximation ratio two.

Theorem 32. *Algorithm 3* Smallest Staircases *computes a Manhattan network for P with total length at most twice the length of a minimum Manhattan network for P.*

Proof. Our strategy of the proof is to partition the plane into two areas and to compare the length of a minimum Manhattan network in each area separately. The first area we consider is the neighboring point area $\mathcal{N}$ defined by x- or y-neighboring points. After Phase I, by Theorem 31

the length of the line segments in MN is at most twice the length of a minimum Manhattan network inside $\mathcal{N}$. The remaining area is the area $\mathbb{R}^2 \setminus \mathcal{N}$ which is the union of the interiors of all staircase areas defined by line segments of MN in Phase I, plus the area which lies outside the Pareto envelope. At this, each staircase area is accounted as an open area. We are allowed to count the line segments added inside the staircase areas independently of the boundary segments only if we know that each staircase area is smallest possible with respect to the neighboring point area $\mathcal{N}$. More precisely, we maximize the area considered by the line segments of the first phase of the algorithm and minimize the staircase areas. A minimum Manhattan network contains for two x-neighboring points at least the length of one y-connection. The same holds for y-neighboring points and x-connections. We do not know where this x- and y-connections lie. To count the segments inserted in the second phase of the algorithm independently, we must choose the line segments in Phase I in such a way that they are best for the staircase areas (such that the areas are smallest concerning these segments). On account of this, we call Procedure 5 Make Smallest Sequence (see Figure 19 (a) and (b)) and Procedure 6 Make Smallest Base. By Lemma 30 after calling Make Smallest Sequence and Make Smallest Base all staircase boundaries are smallest staircase boundaries. Thus, for a staircase sequence $(v_1, \ldots, v_k)$ with base points b^x and b^y, let $\mathcal{A}_{app}$ be the staircase area of the boundary computed by Algorithm 3 in Phase I. The minimum Manhattan network MMN contains (b^x, v_1)- and (b^y, v_k)-paths and shortest (v_i, v_{i+1})-paths, $1 \leq i < k$, inside the neighboring point area $\mathcal{N}$ constituting a staircase boundary B_{opt} inside $\mathcal{N}$. Let $\mathcal{A}_{opt}$ be the staircase area of B_{opt}. After Phase I we get $\mathcal{A}_{app} \subseteq \mathcal{A}_{opt}$.

By the property $\mathcal{A}_{app} \subseteq \mathcal{A}_{opt}$ and since $\mathcal{A}_{app} \cap \mathcal{N} = \emptyset$ holds and since we use only line segments of $MMN \cap \mathcal{N}$ to justify segments inserted in Phase I, we partitioned the problem in two areas which can be considered separately. By Theorem 23 we can compute a Manhattan network for staircases with approximation ratio two inside the staircase area. We can bound the overall approximation ratio of our algorithm by

$$\begin{aligned} |MN_{app}| &= |MN_{app} \cap (\mathbb{R}^2 \setminus \mathcal{N})| + |MN_{app} \cap \mathcal{N}| \\ &\leq 2 \cdot |MMN \cap (\mathbb{R}^2 \setminus \mathcal{N})| + 2 \cdot |MMN \cap \mathcal{N}| \\ &\leq 2 \cdot |MMN|. \end{aligned}$$

□

Last, we want to prove the running time of our algorithm.

Theorem 33. *Algorithm 3* Smallest Staircases *has running time $O(n \log n)$ for n points.*

Proof. First of all we must sort the points of P. This can be done in time $O(n \log n)$. By Lemma 20 the running time of step 1 is $O(n \log n)$. The sweeps performed in steps 2, 3 and 4 each takes time $O(n)$. To find all staircases we simply have to assign to each point its two base points by Definition 1. Thus we can find all staircases in time $O(n)$. The running time to compute a 2-approximation for a Manhattan network of a staircase given the staircase boundary is $O(k \log k)$ for k sequence points by Theorem 24. Each sequence point can only belong to at most four different staircases, thus we get a total running time of $O(n \log n)$ to compute Manhattan networks for all staircases.

This yields the total running time for the algorithm of $O(n \log n)$. □

References

[1] M. Benkert, A. Wolff, F. Widmann and T. Shirabe, The minimum Manhattan network problem: Approximations and exact solutions, Computational Geometry: Theory and Applications 35 (2206), no. 3, 188-208.

[2] G. Chalmet, L. Francis and A. Kolen, Finding efficient solutions for rectilinear distance location problems efficiently, European Journal of Operational Research 6 (1981), 117-124.

[3] V. Chepoi and K. Nouioua and Y. Vaxès, A rounding algorithm for approximating minimum Manhattan networks, Theoretical Computer Science 390 (2008), no. 1, 56-69.

[4] F.Y.L. Chin and Z. Guo and H. Sun, Minimum Manhattan network is NP-complete, Proceedings of the 25th annual symposium on Computational geometry (SCG 09) (2009), 393-402.

[5] L. P. Chew, There are planar graphs almost as good as the complete graph, Journal of Computer and System Sciences 39 (1989), no. 2, 205-219.

[6] D. Eppstein, Spanning trees and spanners, Handbook of Computational Geometry (Jörg-Rudiger Sack and Jorge Urrutia, eds.), Elsevier (2000), pp. 425-461.

[7] B. Fuchs and A. Schulze, A simple 3-approximation of minimum Manhattan networks, Technical report zaik2008-570, Universität Köln, submitted

[8] J. Gudmundsson, O. Klein, C. Knauer and M. Smid, Small Manhattan networks and algorithmic applications for the earth mover's distance, Proceedings of the 23th European Workshop on Computational geometry (SCG 07) (2007), 174-177.

[9] J. Gudmundsson, C. Levcopoulos and G. Narasimhan, Approximating a minimum Manhattan network, Nordic Journal of Computing, 8 (2001), no. 2, 219-232.

[10] Z. Guo, H. Sun, and H. Zhu, A fast 2-approximation algorithm for the minimum Manhattan network problem, Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management, Springer-Verlag (2008), 212-223.

[11] Z. Guo, H. Sun and H. Zhu, Greedy Construction of 2-Approximation Minimum Manhattan Network, Proceedings of the 19th International Symposium on Algorithms and Computation (ISAAC 09), Springer Verlag (2008), 4-15.

[12] R. Kato, K. Imai and T. Asano, An improved algorithm for the minimum Manhattan network problem, Proceedings of the 13th International Symposium on Algorithms and Computations, Lecture Notes in Computer Science, vol. 2518, Springer Verlag (2002), 344-356.

[13] X. Mu noz and S. Seibert and W. Unger, The minimal Manhattan network problem in three dimensions, Proceedings of the 3rd International Workshop on Algorithms and Computation (WALCOM09), Springer Verlag (2009), 369-380.

[14] G. Narasimhan and M. Smid, Geometric spanner networks, Cambridge University Press, New York, NY, USA (2007).

[15] K. Nouioua, Enveloppes de Pareto et Réseaux de Manhattan: caractérisations et algorithmes, Ph.D. thesis, Université de la Méditerranée (2005).

[16] F. Lam, M. Alexandersson and L. Pachter, Picking alignments from (Steiner) trees, Proceedings of the sixth annual international conference on Computational biology (RECOMB 02), ACM, New York (2002), 246-253.

[17] S. Seibert and W. Unger, A 1.5-approximation of the minimal Manhattan network problem, Proceedings 16th International Symposium on Algorithms and Computation, Lecture Notes in Computer Science, vol. 3827, Springer-Verlag (2005), 246-255.

[18] R.E. Wendell, A.P. Hurter and T.J. Lowe, Efficient points in location theory, AIEE Transactions 9 (1971), 314-321.